THIRD ANNUAL

2003 Microsoft® **STRATEGIC ARCHITECT FORUM**

for Customers

Solution Architecture for the Service-Oriented Enterprise

Keith Short, Architect
Visual Studio, Enterprise Tools
Microsoft Corporation

Summary

- Building service-oriented applications is a complex undertaking
- We can improve the underlying frameworks and languages to help
- But modeling languages, tools, and patterns have an important role to play
- We can build effective modeling tools if we follow some simple rules

Agenda

- Complexity in Connected Systems
- Using Abstraction to Reduce Complexity
- Domain-Specific Modeling Languages (DSLs) and Patterns
- Scenario 1: Web Services and Design for Operations
- Scenario 2: Implementing Business Processes
- Strategy and Tools for Microsoft Visual Studio® “Whidbey”

Agenda

- Complexity in Connected Systems
- Using Abstraction to Reduce Complexity
- Domain-Specific Modeling Languages and Patterns
- Scenario 1: Web Services and Design for Operations
- Scenario 2: Implementing Business Processes
- Microsoft Strategy and Tools for Visual Studio “Whidbey”

New Business Requirements

- Automate Business Processes
- Value Chains with Partners
- Customer and Supplier Self Serv
- Improve ROI of IT
- Lower Operating Costs
- Cut Time to Market

New Application Architecture

Automate Business Processes
Improve ROI of IT
Lower Operating Costs
Cut Time to Market

- Service-Oriented Architecture
- Web Services, Messaging, Contrac
- Security
- Caching and State Management
- Deployment Policies and Constrain
- Heterogeneous Platforms

Evolving Platform Technology

Automate Business Processes
Improve ROI of IT
Lower Operating Costs

- Application Servers and Standar
- Schemas and Databases
- Code and Attributes
- Configuration Files
- Project Structures
- Physical Server Configurations

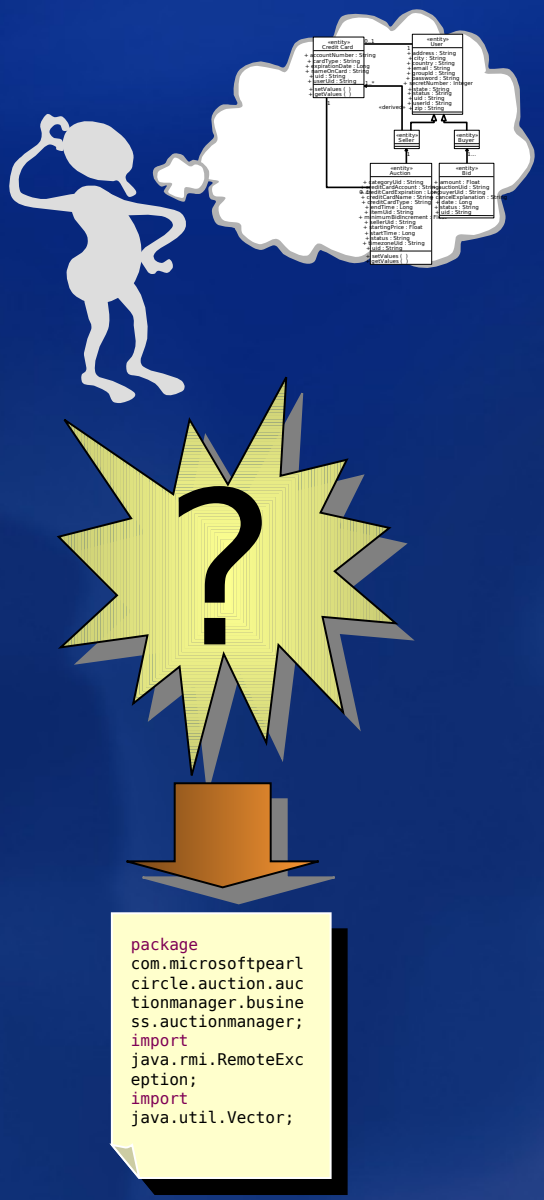
Complexity

- Automate Business Processes
- Value Chains with Partners
- Customer and Supplier Self Service
- Improve ROI of IT
- Lower Operating Costs
- Cut Time to Market

- Service-Oriented Architecture
- Web Services, Messaging, Contracts
- Security
- Caching and State Management
- Deployment Policies and Constraints
- Heterogeneous Platforms

- Application Servers and Standards
- Schemas and Databases
- Code and Attributes
- Configuration Files
- Project Structures
- Physical Server Configurations

Where Does Complexity Arise?

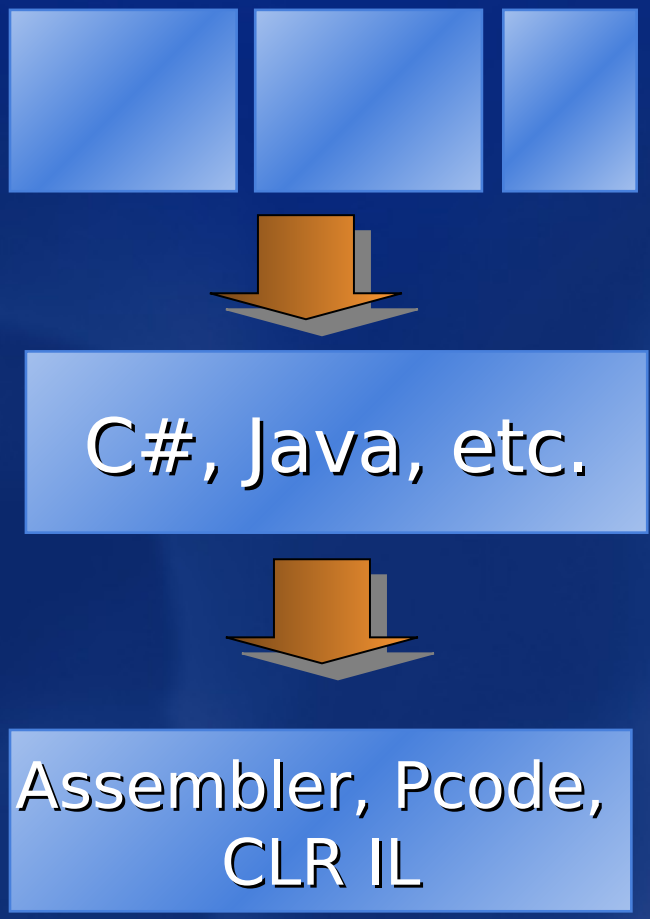


- High-Level Concepts
- Low-Level Implementation Languages

Agenda

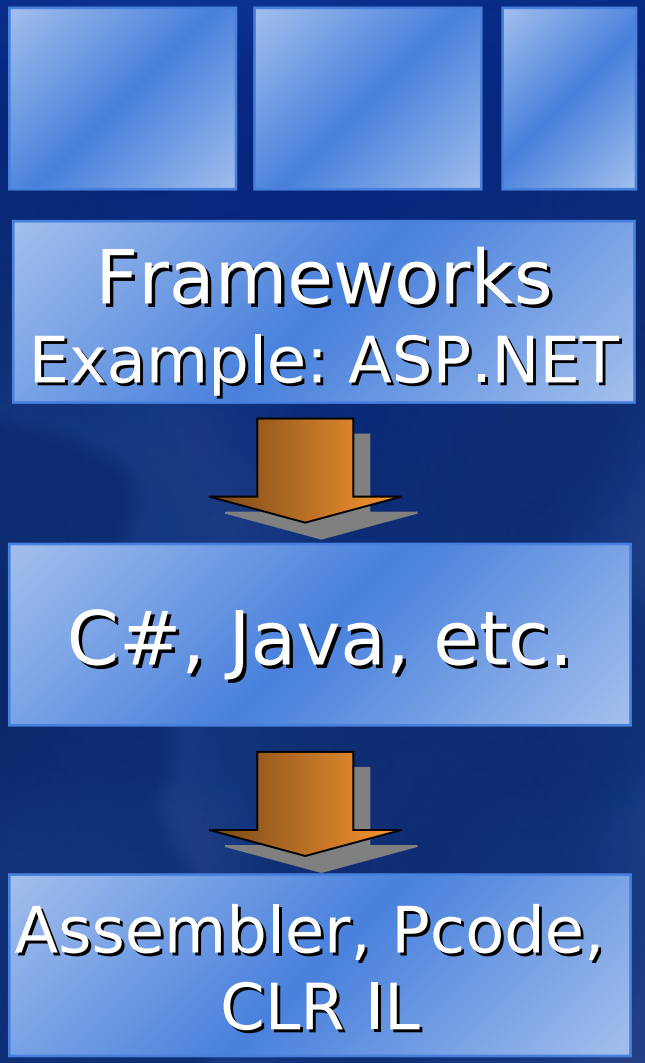
- Complexity in Connected Systems
- Using Abstraction to Reduce Complexity
- Domain-Specific Modeling Languages and Patterns
- Scenario 1: Web Services and Design for Operations
- Scenario 2: Implementing Business Processes
- Microsoft Strategy and Tools for Visual Studio “Whidbey”

Abstraction Reduces Complexity



- Domain-Specific Abstractions
- General-Purpose Abstractions
- Bytecode or Machine Languages

Frameworks Reduce Translation Complexity



- Domain-Specific Abstractions
- Domain-Specific Frameworks
- General-Purpose Abstractions
- Bytecode or Machine Languages

Agenda

- Complexity in Connected Systems
- Using Abstraction to Reduce Complexity
- Domain-Specific Modeling Languages and Patterns
- Scenario 1: Web Services and Design for Operations
- Scenario 2: Implementing Business Processes
- Microsoft Strategy and Tools for Visual Studio “Whidbey”

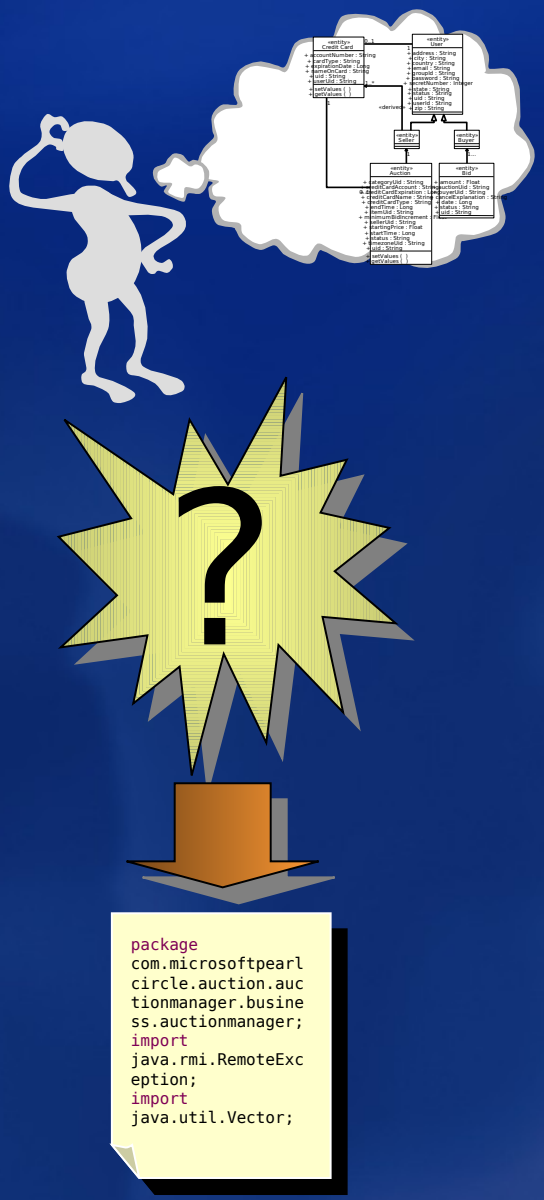
Domain-Specific Modeling Language

- Domain-Specific Abstractions
- Narrow Scope
- Graphical
- Highly Expressive
- Coarse-Grained Abstractions
- May Be Translated into:
  - General-purpose programming languages
  - Specific framework completion
  - Other modeling languages

Understanding Web Services

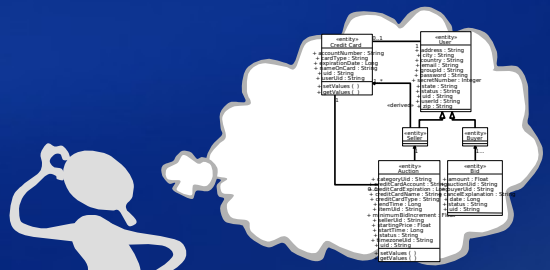
- Activities <-> Services
- Web Services
- Messages
- Contracts
- Sequences
- Security
- Reuse
- Wire-Level Protocols
- etc.

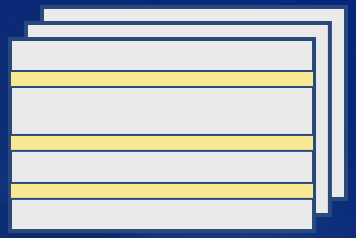
Where Does Complexity Arise?



- High-Level Concepts
  - Activities
  - Messages
  - Security
  - Reuse
  - Services
  - Contracts
- Low-Level Implementation Languages

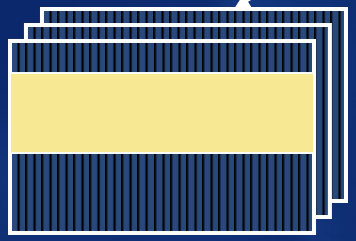
Scattered Concepts ...





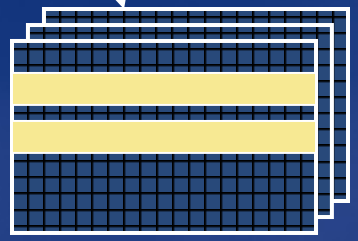
Deployment Policies and Packages

Project Files and Templates



.asmx Files

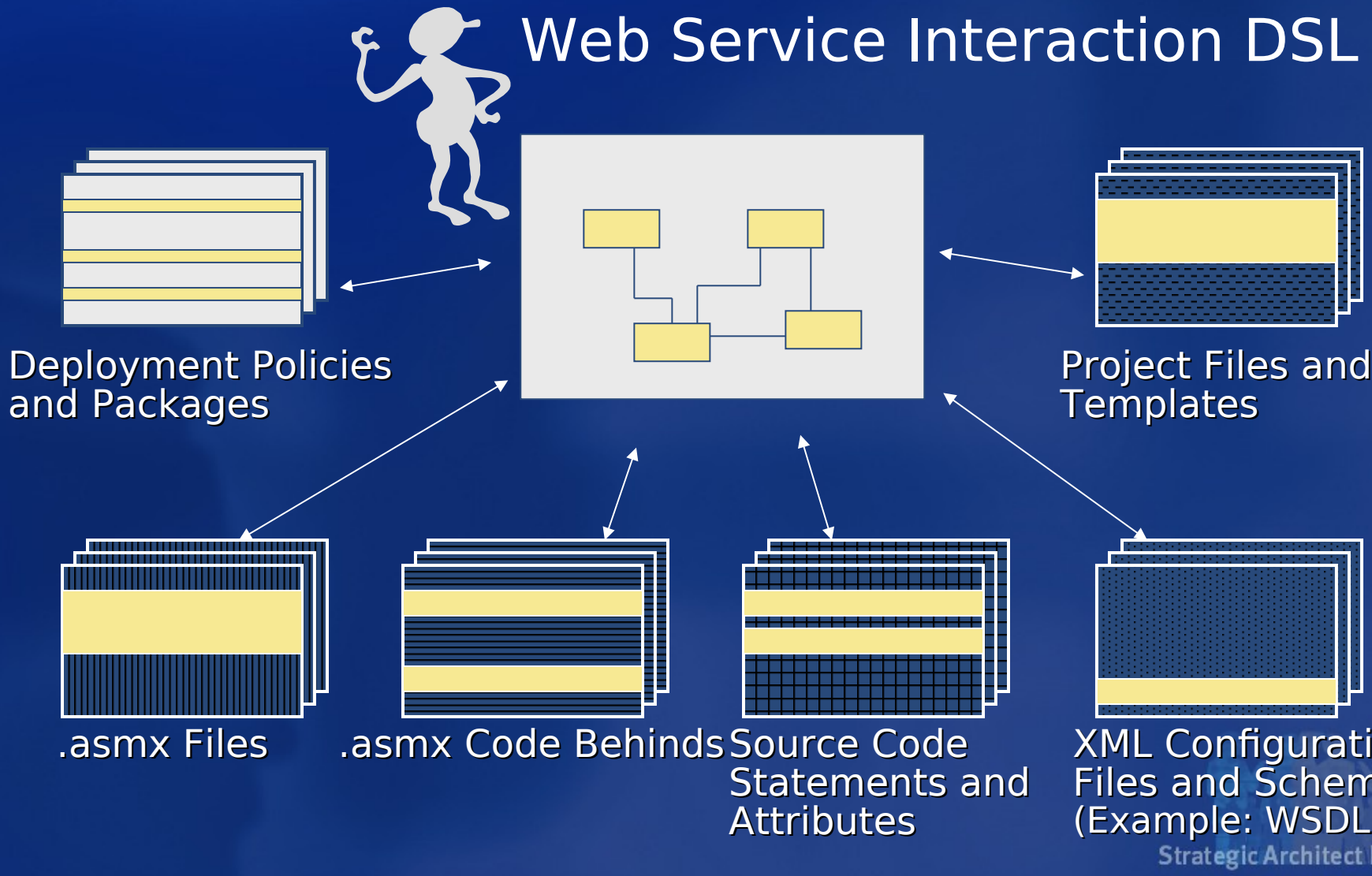
.asmx Code Behinds



Source Code Statements and Attributes

XML Configuration Files and Schemas (Example: WSDL Files

…Replaced by an Holistic View



Need a Designer Infrastructure

- For a Specific Aspect of Overall Development
  - Define a metamodel for the abstractions
  - Define a graphical notation
  - Define a synchronization engine

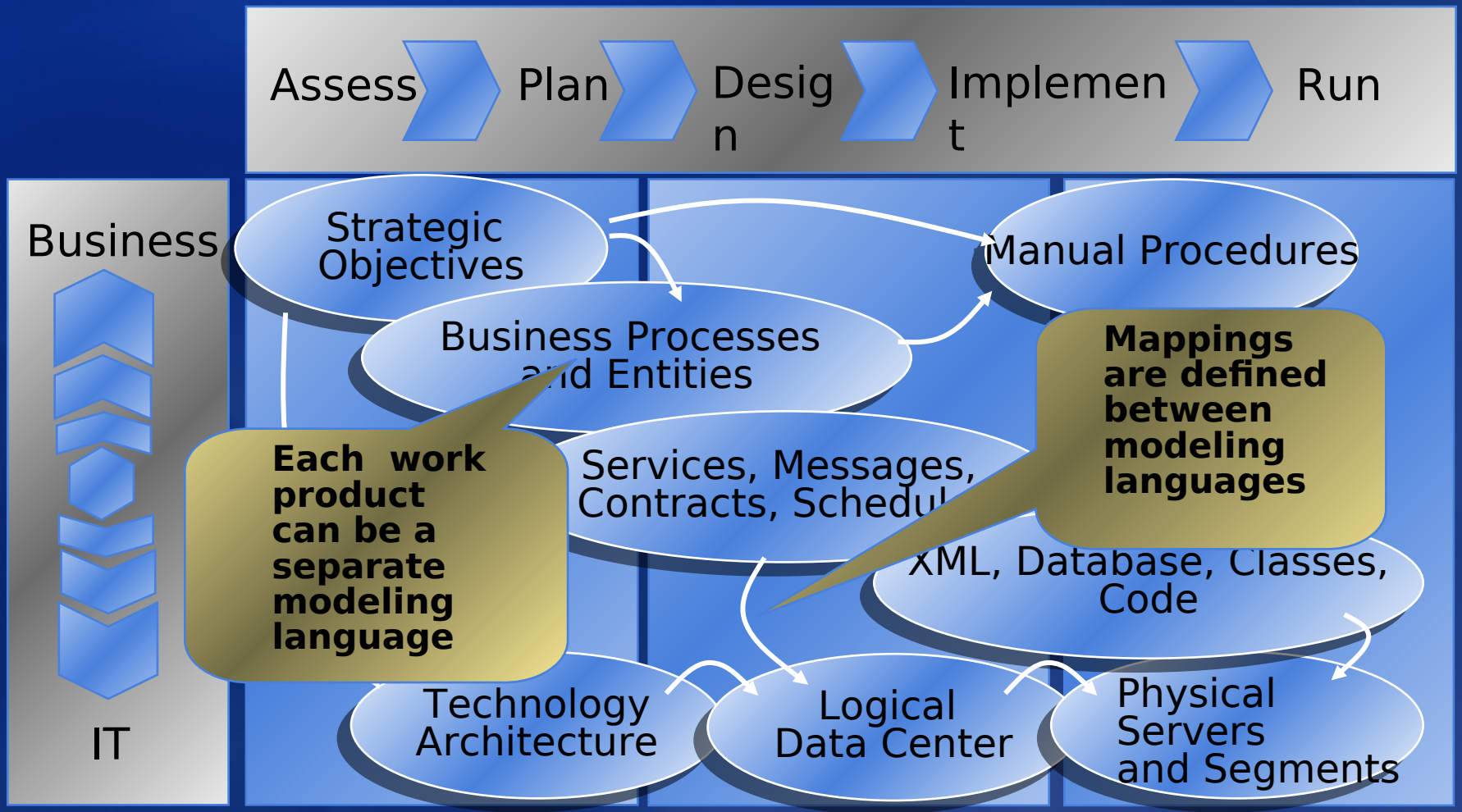
Complexity

- Automate Business Processes
- Value Chains with Partners
- Customer and Supplier Self Service
- Improve ROI of IT
- Lower Operating Costs
- Cut Time to Market

- Service-Oriented Architecture
- Web Services, Messaging, Contracts
- Security
- Caching and State Management
- Deployment Policies and Constraints
- Heterogeneous Platforms

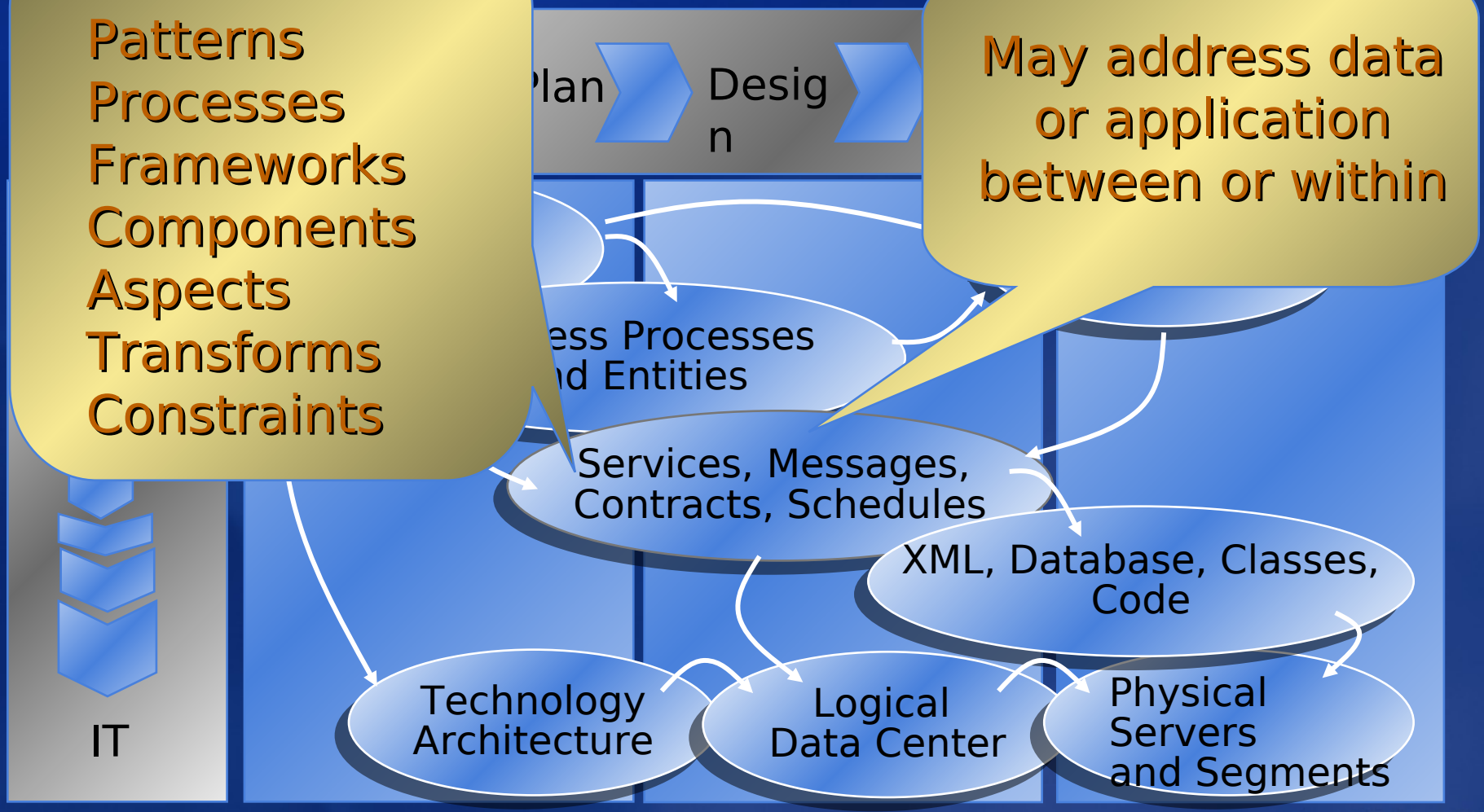
- Application Servers and Standards
- Schemas and Databases
- Code and Attributes
- Configuration Files
- Project Structures
- Physical Server Configurations

Simplify Using DSLs



What's Inside a Viewpoint?

DSLs
Patterns
Processes
Frameworks
Components
Aspects
Transforms
Constraints

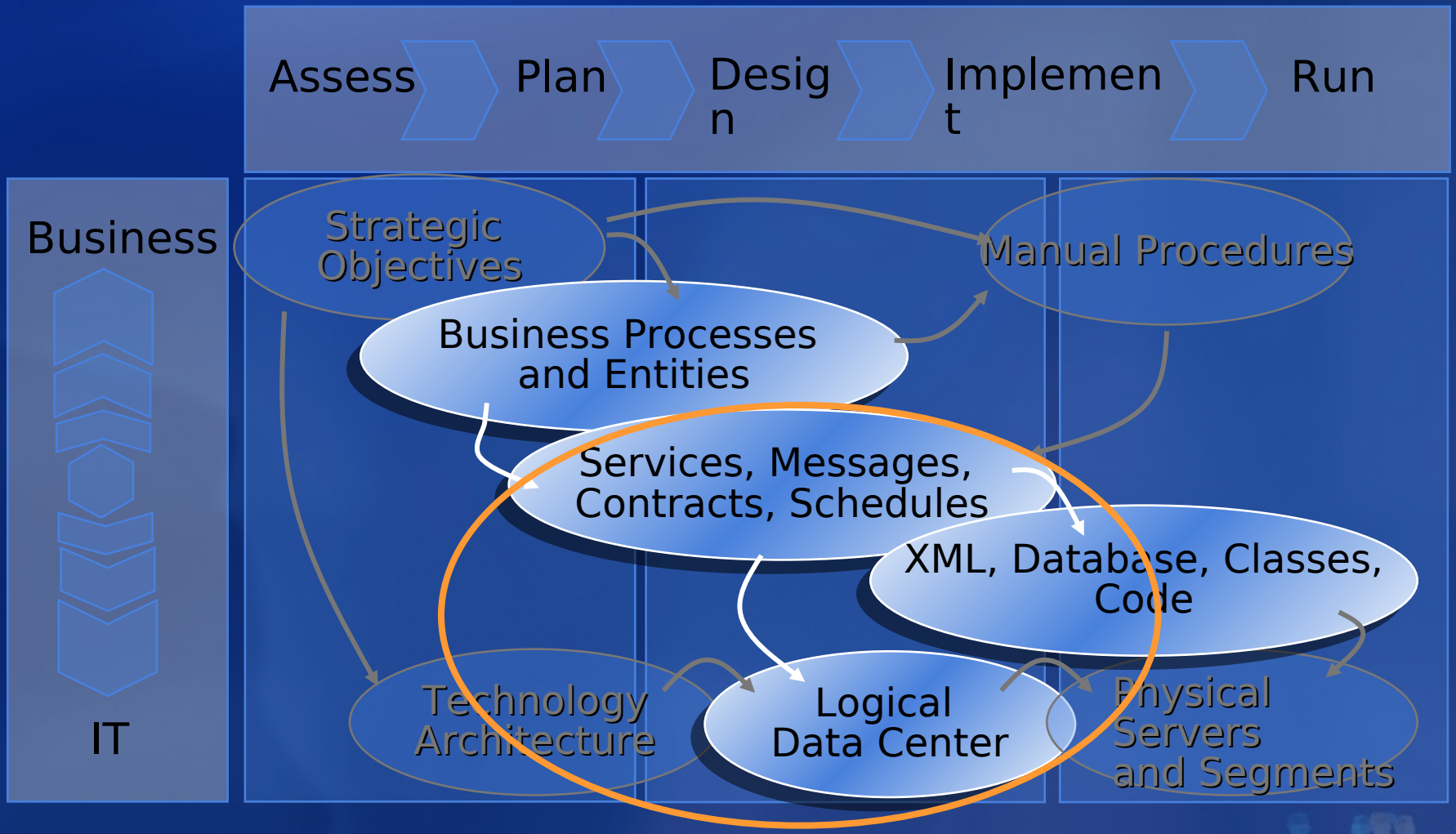


May address data or application between or within

Agenda

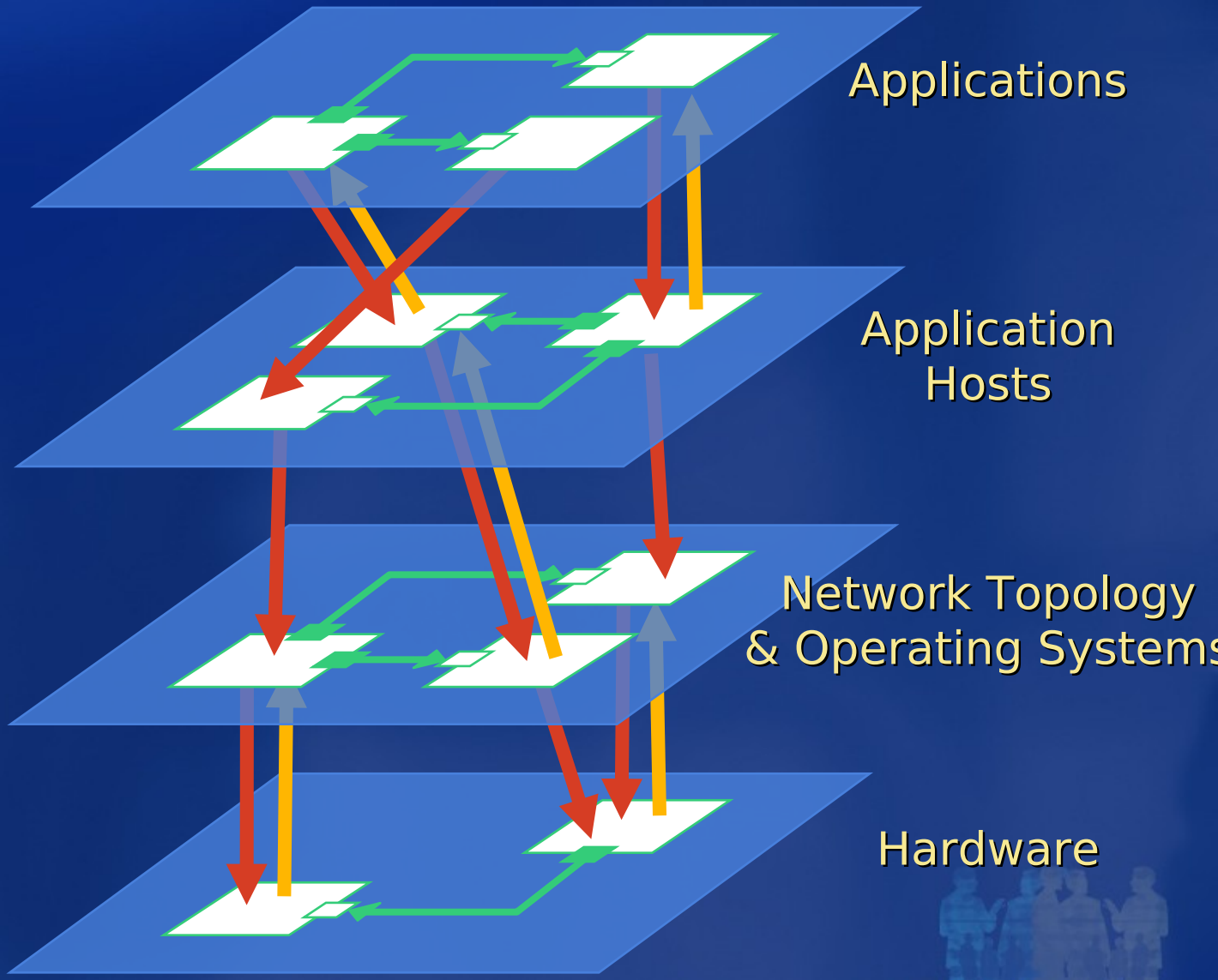
- Complexity in Connected Systems
- Using Abstraction to Reduce Complexity
- Domain-Specific Modeling Languages and Patterns
- Scenario 1: Web Services and Design for Operations
- Scenario 2: Implementing Business Processes
- Microsoft Strategy and Tools for Visual Studio “Whidbey”

Example 1


Assess
Plan
Design
Implement
Run
Business
IT
Strategic Objectives
Manual Procedures
Business Processes and Entities
Services, Messages, Contracts, Schedules
XML, Database, Classes, Code
Technology Architecture
Logical Data Center
Physical Servers and Segments

Dynamic Systems Initiative (DSI)

System Definition Model



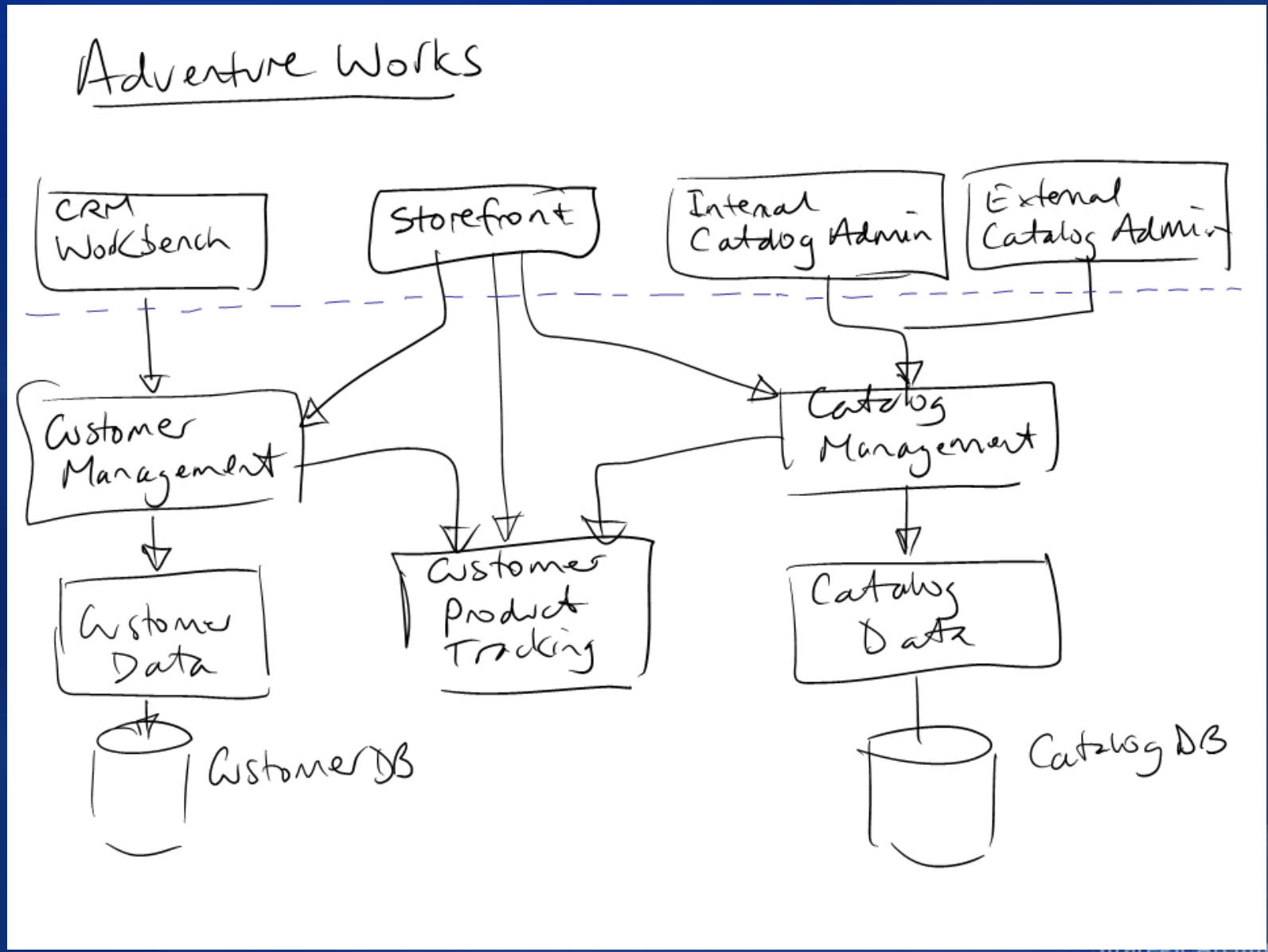
“Whidbey” Solutions…

An Application Designer

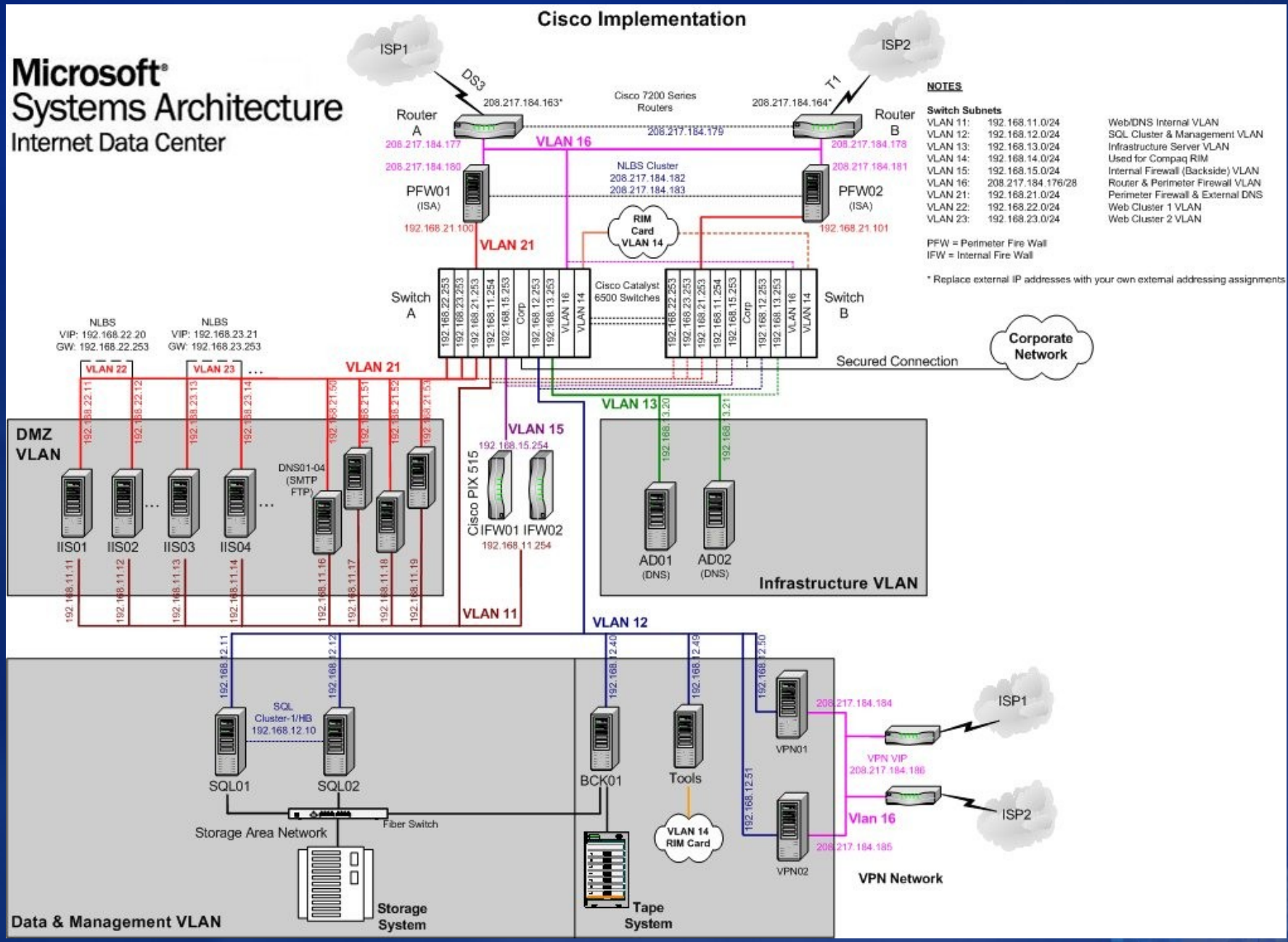
System Definition Model

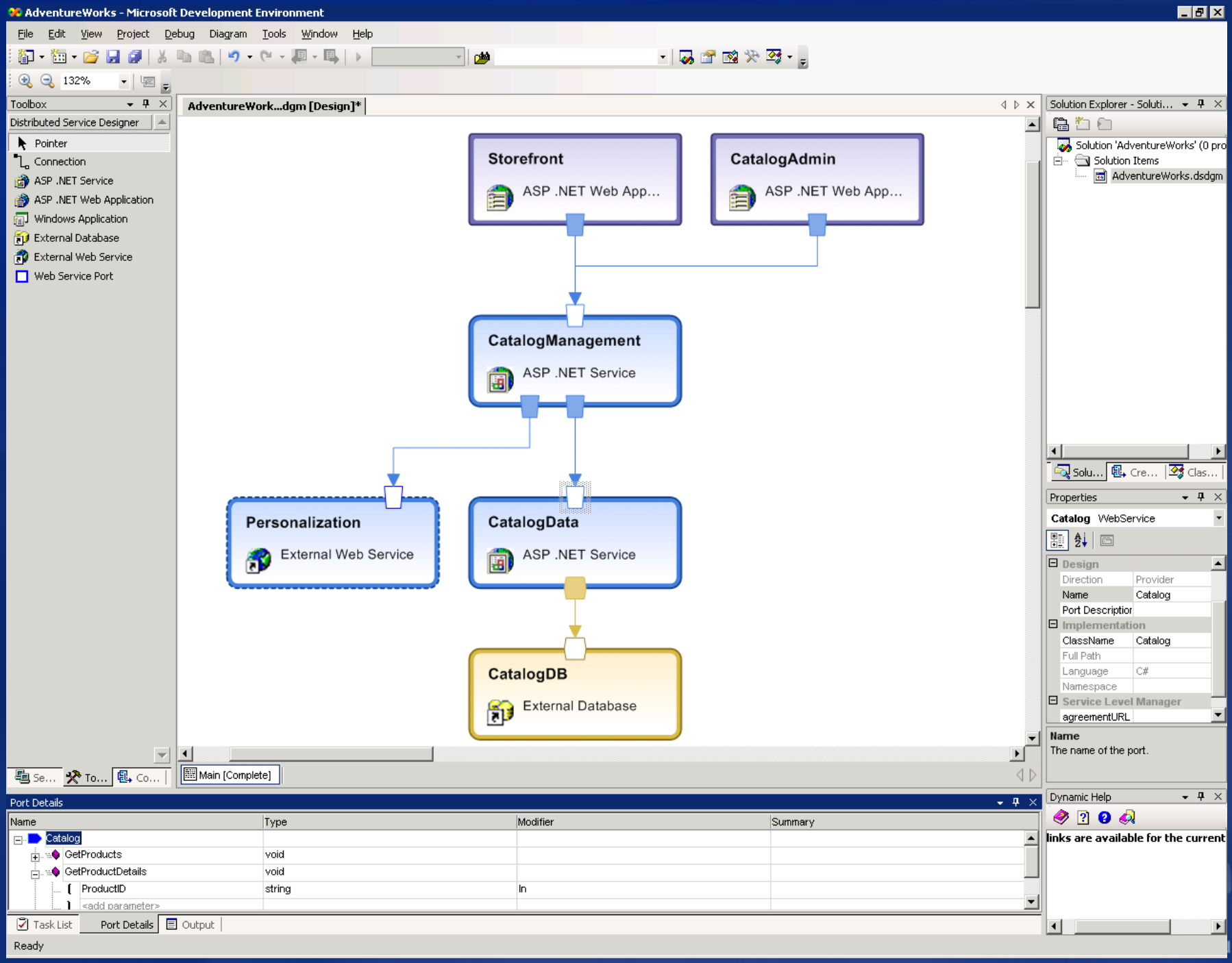
A Logical Data Center Designer

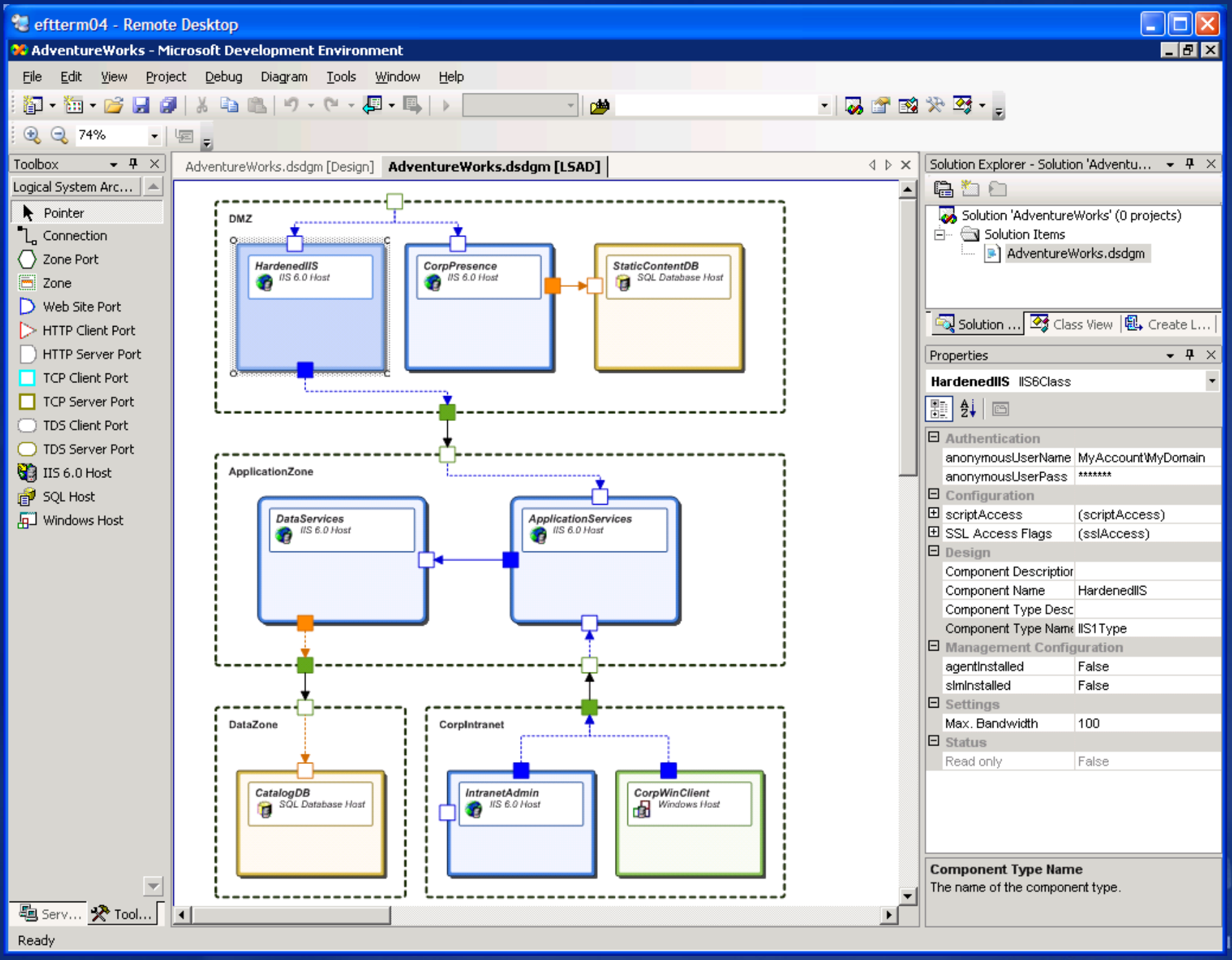
A DSL in Use Today

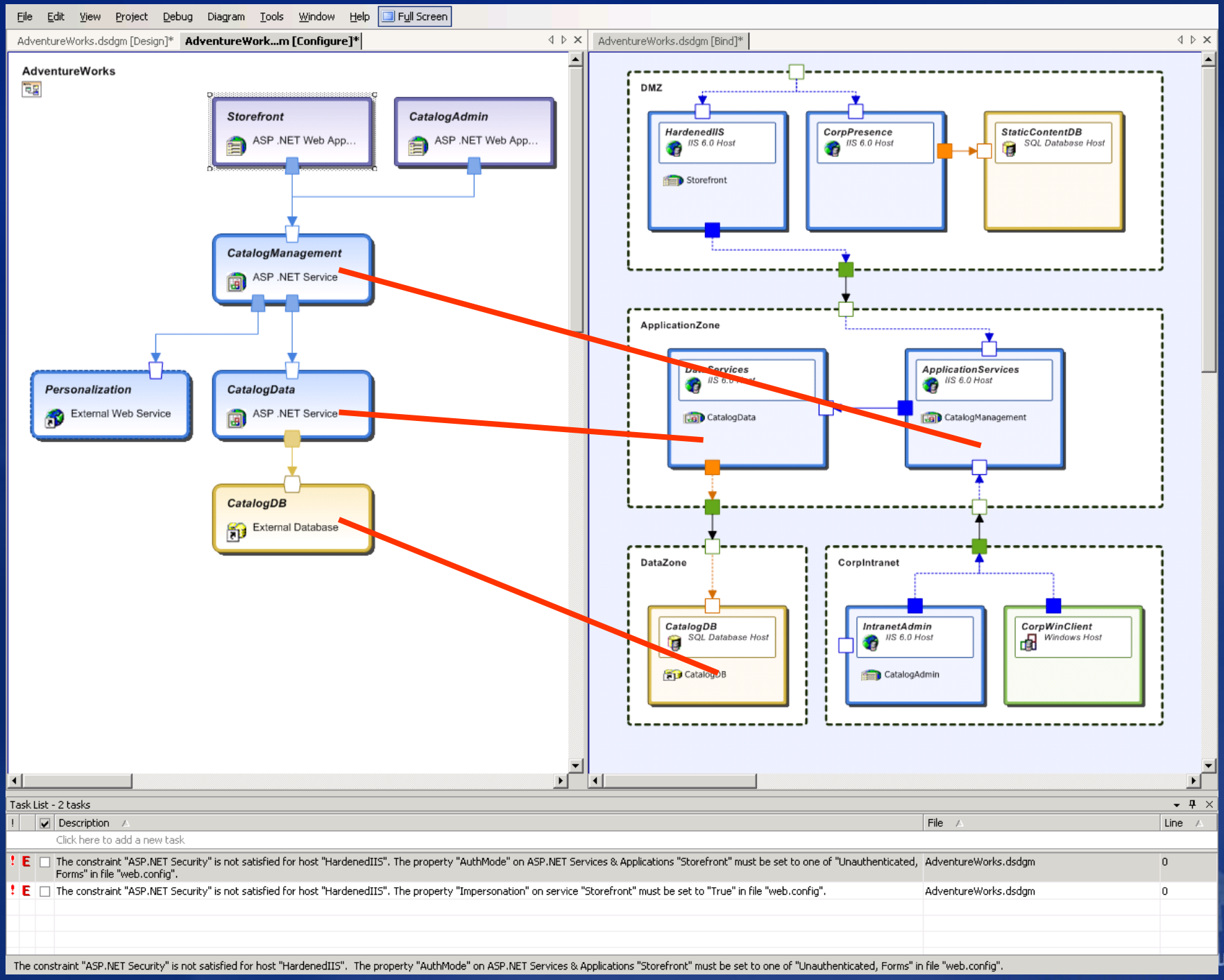


Microsoft® Systems Architecture
Internet Data Center




AdventureWorks - Microsoft Development Environment
File Edit View Project Debug Diagram Tools Window Help
Toolbox
Distributed Service Designer
Pointer
Connection
ASP .NET Service
ASP .NET Web Application
Windows Application
External Database
External Web Service
Web Service Port
AdventureWork...dgm [Design]*
Storefront
ASP .NET Web App...
CatalogAdmin
ASP .NET Web App...
CatalogManagement
ASP .NET Service
Personalization
External Web Service
CatalogData
ASP .NET Service
CatalogDB
External Database
Main [Complete]
Solution Explorer - Soluti...
Solution 'AdventureWorks' (0 pro
Solution Items
AdventureWorks.dsdgm
Properties
Catalog WebService
Design
Direction Provider
Name Catalog
Port Description
Implementation
ClassName Catalog
Full Path
Language C#
Namespace
Service Level Manager
agreementURL
Name
The name of the port.
Port Details
Name Type Modifier Summary
Catalog
GetProducts void
GetProductDetails void
ProductID string In
<add parameter>
Dynamic Help
links are available for the current
Task List Port Details Output
Ready


efftterm04 - Remote Desktop
AdventureWorks - Microsoft Development Environment
File Edit View Project Debug Diagram Tools Window Help
74%
Toolbox
Logical System Arc...
Pointer
Connection
Zone Port
Zone
Web Site Port
HTTP Client Port
HTTP Server Port
TCP Client Port
TCP Server Port
TDS Client Port
TDS Server Port
IIS 6.0 Host
SQL Host
Windows Host
AdventureWorks.dsdgm [Design]
AdventureWorks.dsdgm [LSAD]
DMZ
HardenedIIS
IIS 6.0 Host
CorpPresence
IIS 6.0 Host
StaticContentDB
SQL Database Host
ApplicationZone
DataServices
IIS 6.0 Host
ApplicationServices
IIS 6.0 Host
DataZone
CatalogDB
SQL Database Host
CorpIntranet
IntranetAdmin
IIS 6.0 Host
CorpWinClient
Windows Host
Solution Explorer - Solution 'Adventu...
Solution 'AdventureWorks' (0 projects)
Solution Items
AdventureWorks.dsdgm
Solution ...
Class View
Create L...
Properties
HardenedIIS IIS6Class
Authentication
anonymousUserName MyAccount\MyDomain
anonymousUserPass *******
Configuration
scriptAccess (scriptAccess)
SSL Access Flags (sslAccess)
Design
Component Description
Component Name HardenedIIS
Component Type Desc
Component Type Name IIS1Type
Management Configuration
agentInstalled False
slmInstalled False
Settings
Max. Bandwidth 100
Status
Read only False
Component Type Name
The name of the component type.
Serv...
Tool...
Ready


File Edit View Project Debug Diagram Tools Window Help Full Screen
AdventureWorks.dsdgm [Design]*
AdventureWork...m [Configure]*
AdventureWorks.dsdgm [Bind]*
AdventureWorks
Storefront
ASP .NET Web App...
CatalogAdmin
ASP .NET Web App...
CatalogManagement
ASP .NET Service
Personalization
External Web Service
CatalogData
ASP .NET Service
CatalogDB
External Database
DMZ
HardenedIIS
IIS 6.0 Host
Storefront
CorpPresence
IIS 6.0 Host
StaticContentDB
SQL Database Host
ApplicationZone
ApplicationServices
IIS 6.0 Host
CatalogManagement
CatalogData
DataZone
CatalogDB
SQL Database Host
CatalogDB
CorpIntranet
IntranetAdmin
IIS 6.0 Host
CatalogAdmin
CorpWinClient
Windows Host
Task List - 2 tasks
Description
File
Line
Click here to add a new task
The constraint "ASP.NET Security" is not satisfied for host "HardenedIIS". The property "AuthMode" on ASP.NET Services & Applications "Storefront" must be set to one of "Unauthenticated, Forms" in file "web.config".
AdventureWorks.dsdgm
0
The constraint "ASP.NET Security" is not satisfied for host "HardenedIIS". The property "Impersonation" on service "Storefront" must be set to "True" in file "web.config".
AdventureWorks.dsdgm
0
The constraint "ASP.NET Security" is not satisfied for host "HardenedIIS". The property "AuthMode" on ASP.NET Services & Applications "Storefront" must be set to one of "Unauthenticated, Forms" in file "web.config".

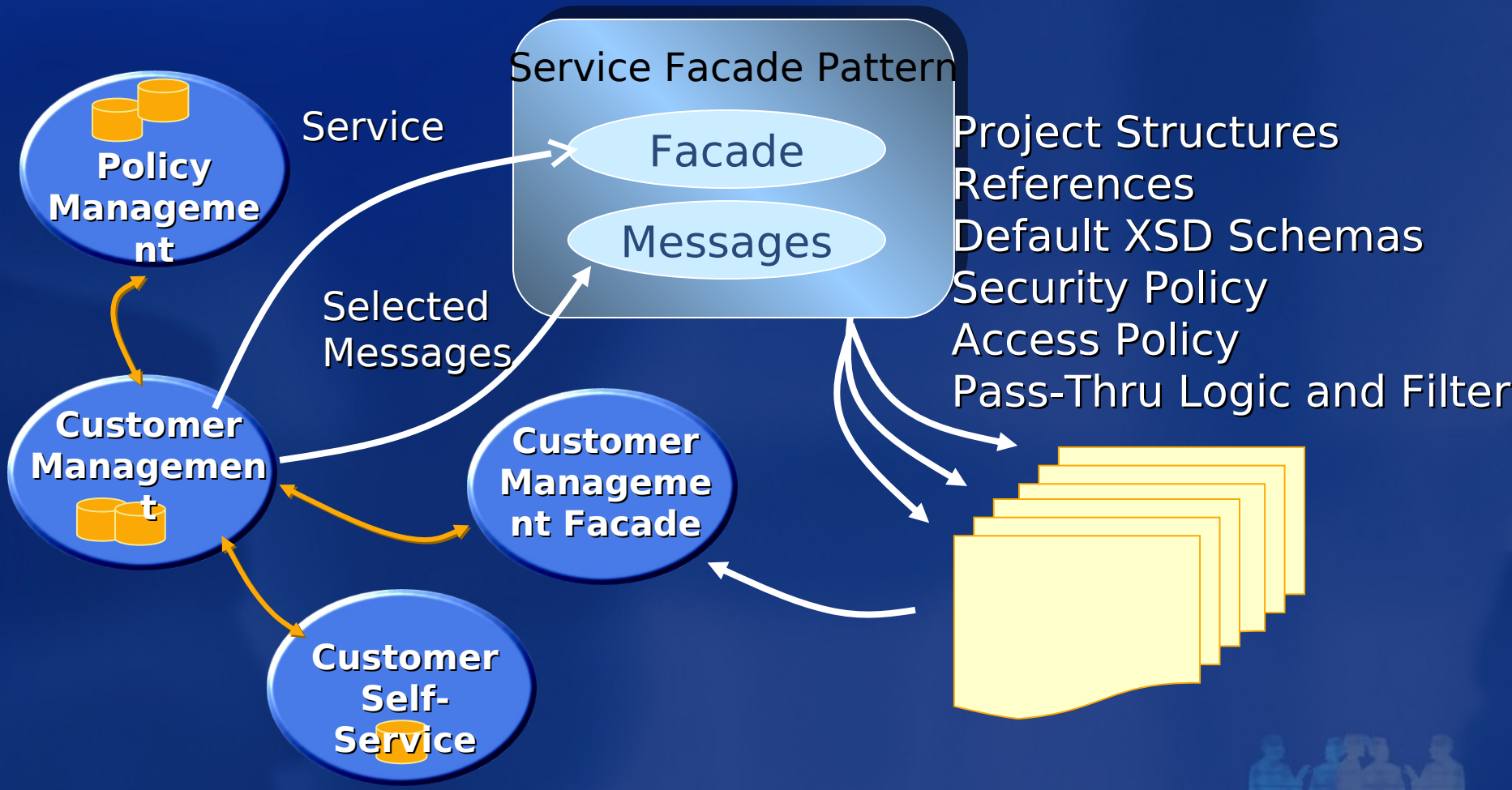
Web Service Interaction

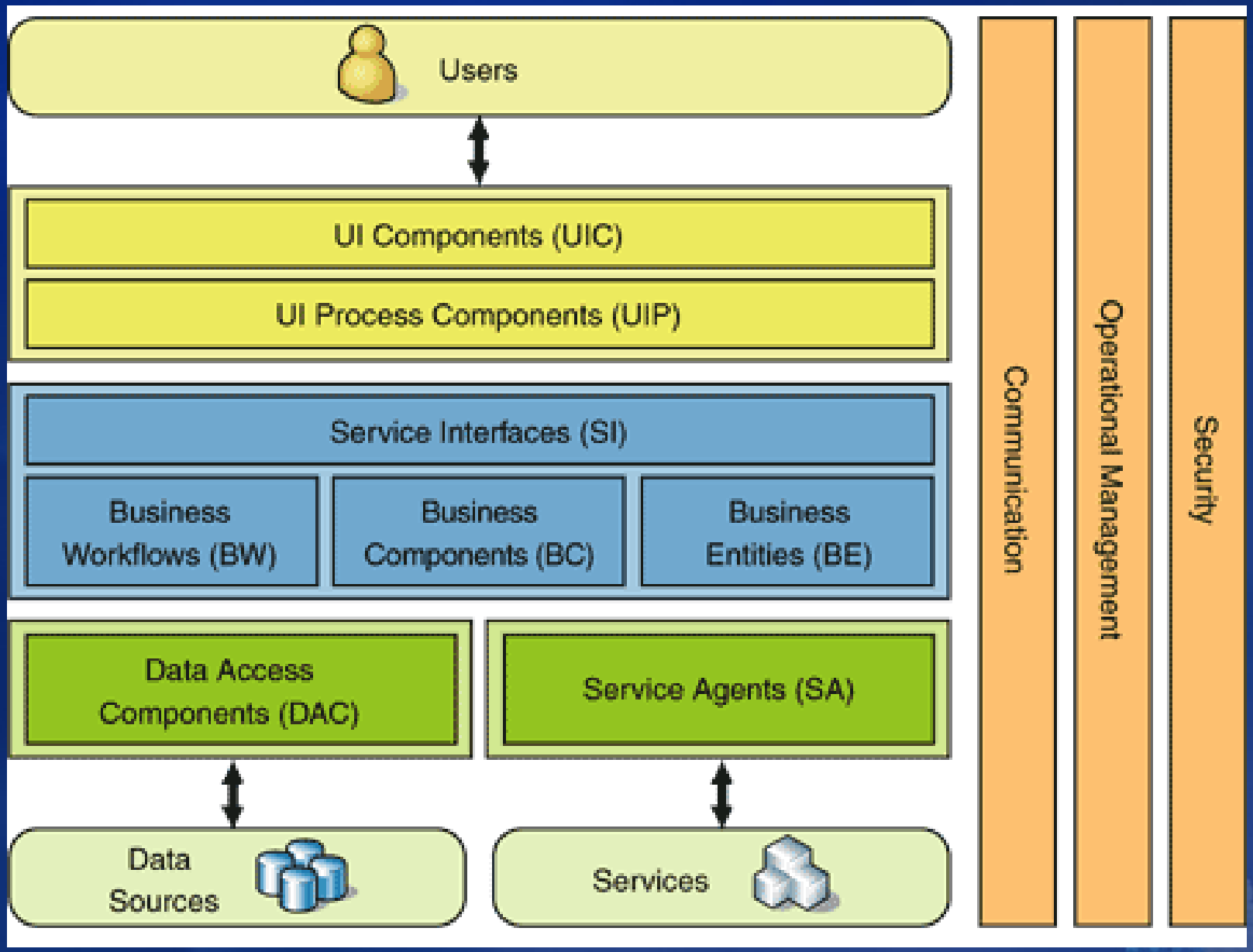
- Patterns
  - Service interface
  - Gateway
  - Façade
  - Layered architecture
- Aspects
  - Transactions
  - Security
- Transformations
  - Validate against deployment infrastructure
  - Transform into implementation artifacts

Logical System Architecture

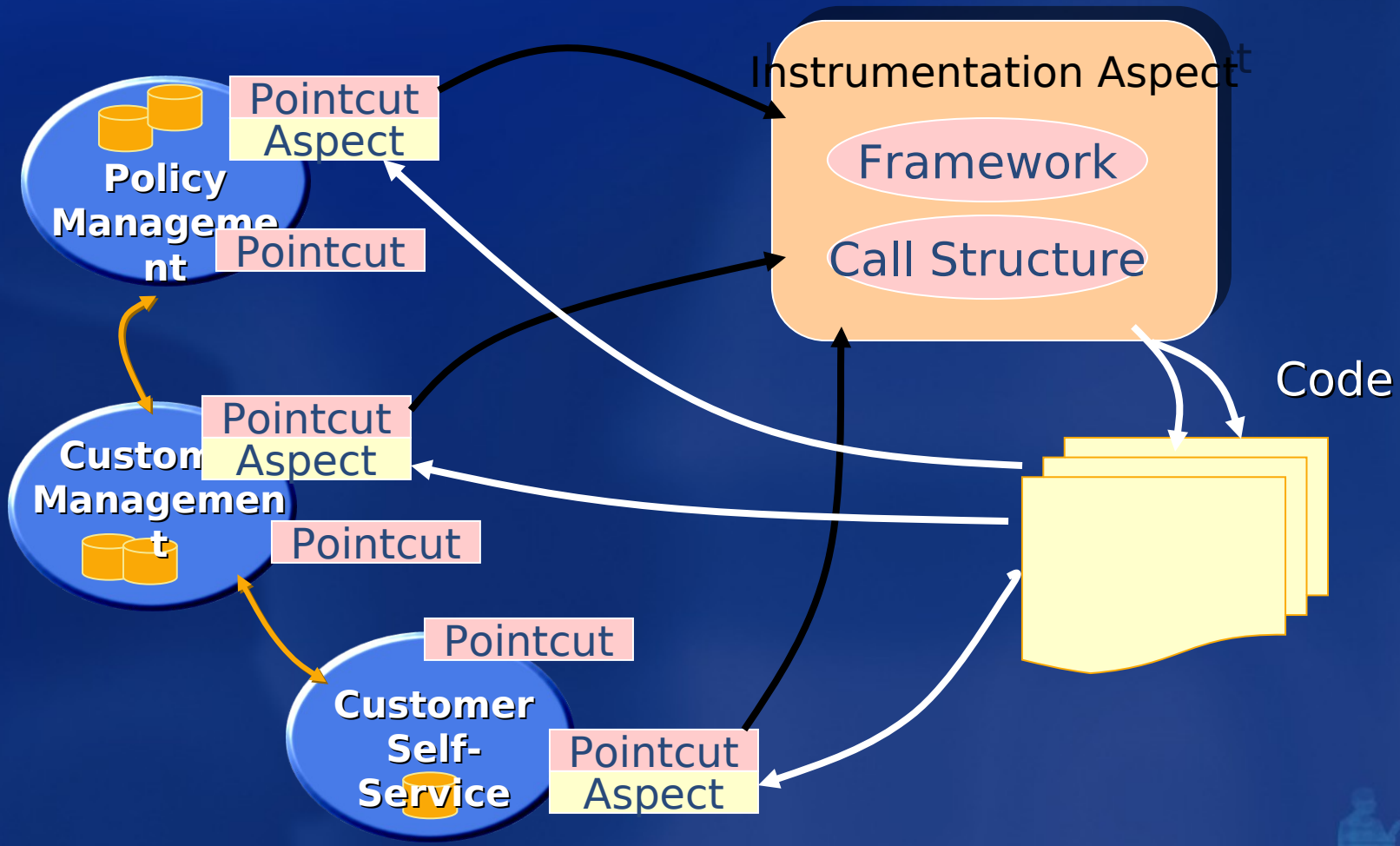
- Logical System Architecture Patterns
  - “Hardened Internet Information Services” (IIS)—metabase settings
  - MSA patterns
    - IDC, EDC
  - Tiered architecture patterns
- Aspects
  - Firewall settings
  - Protocol settings
- Transformations
  - Validate against Web service application design
  - Transform into implementation System Definition Model (SDM)

Applying a Service Façade Pattern




Users
UI Components (UIC)
UI Process Components (UIP)
Service Interfaces (SI)
Business Workflows (BW)
Business Components (BC)
Business Entities (BE)
Data Access Components (DAC)
Service Agents (SA)
Data Sources
Services
Communication
Operational Management
Security

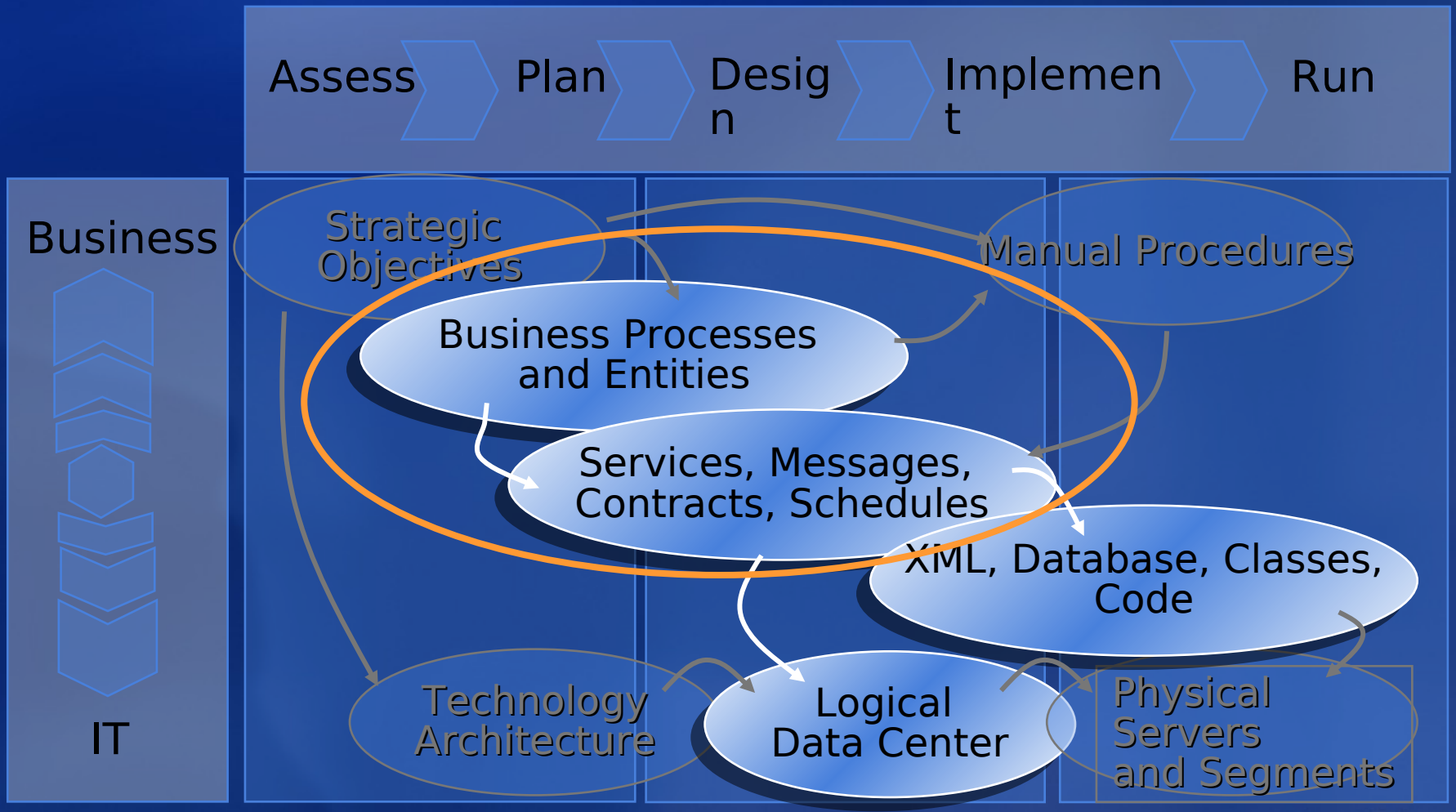
Aspect Weaving: Instrumentation



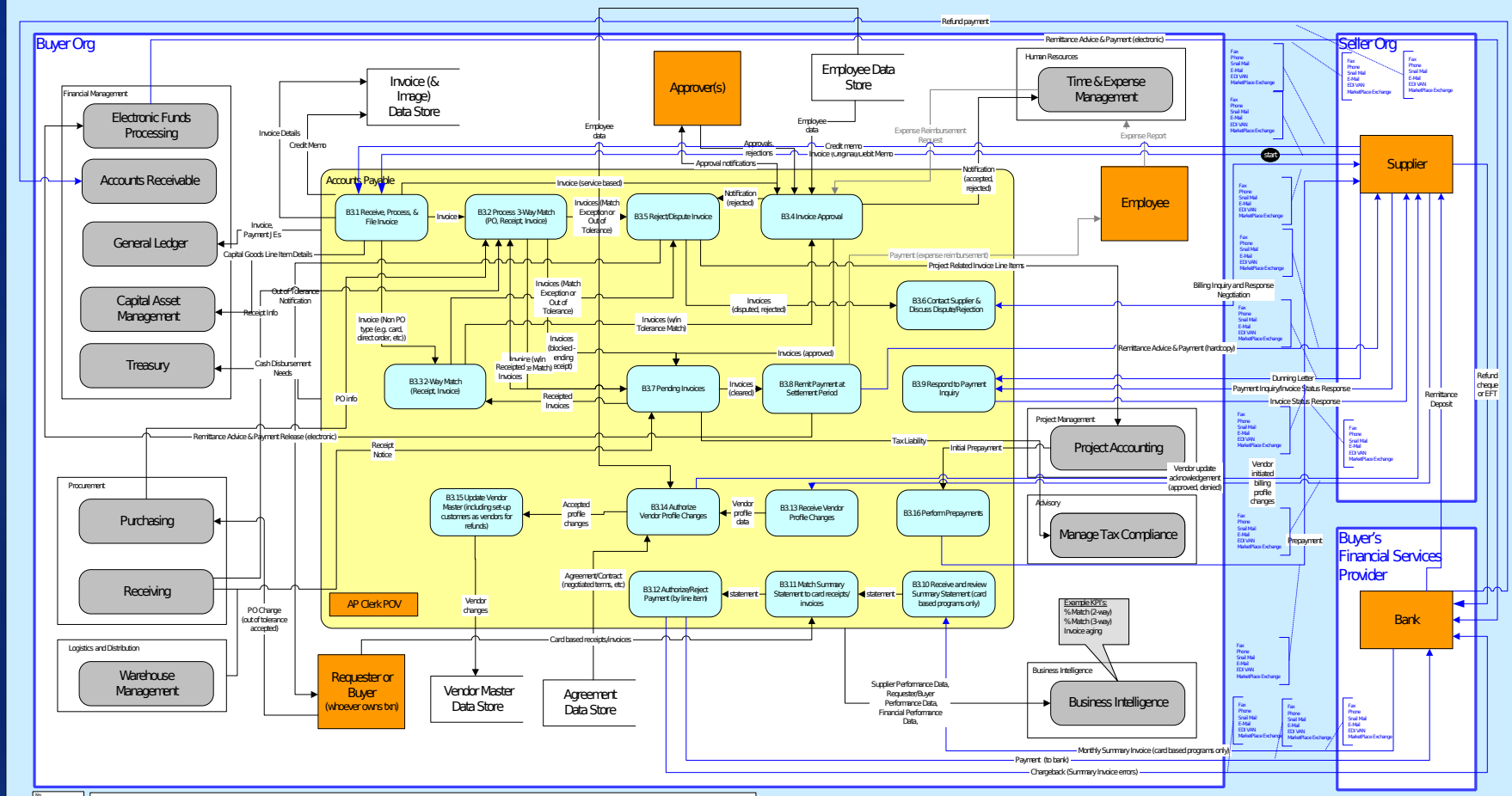
Agenda

- Complexity in Connected Systems
- Using Abstraction to Reduce Complexity
- Domain-Specific Modeling Languages and Patterns
- Scenario 1: Web Services and Design for Operations
- Scenario 2: Implementing Business Processes
- Microsoft Strategy and Tools for Visual Studio “Whidbey”

Scenario 2


Assess
Plan
Design
Implement
Run
Business
IT
Strategic Objectives
Manual Procedures
Business Processes and Entities
Services, Messages, Contracts, Schedules
XML, Database, Classes, Code
Technology Architecture
Logical Data Center
Physical Servers and Segments

Invoice to Payment



Acronyms: AP: Accounts Payable, JEs: Journal Entries
PO: Purchase Order, KPI: Key Performance Indicator

Buyer Org Perspective: Requisition to Payment

B3. Invoice to Payment (AP)

Preconditions: A purchase order has been created and sent to the Seller Organization.

A supplier invoice usually triggers the Invoice to Payment process. In process step B3.1, a supplier submits an invoice to the buyer organization, usually after they've picked, packed, and shipped product to the buyer (although pre-payments sometimes occur). The invoice is reviewed for accuracy and entered into a payables system for matching purposes. This can also trigger the flow of information to Capital Asset management tracking for depreciation purposes, Project Accounting for project financials tracking, Tax for compliance reporting, and the General Ledger to account for liability owed. In process step B3.2, the buying organization's Accounts Payable group/person then can perform a 3-way match between the original PO, the materials Receipt, and the Invoice. Process step B3.3 illustrates how some alternative purchasing models can be accommodated, such as: direct orders, employee expense report submission (which requires payment to the employee) or procurement card (like a credit card) based purchases, where a the requisition and PO is entirely bypassed. Process B3.4 allows for these alternative 3-way matching scenarios , as the invoice is approved and verified before payment is made to the vendor. In the event of match exception or out of tolerance scenarios (where amount invoiced is considerably different than expected), Accounts Payable notifies the requester of over tolerance (which could trigger a PO change) or rejects (disputes) the invoice (B3.5). In process step B3.6, Accounts Payable (or the Buyer) contacts the Supplier to resolve the rejected invoice. If matching and tolerance proceeds as planned, in process step B3.7 the invoice could be held in a pending state until a settlement period has been reached, optimizing working capital. Payment is then remitted to the supplier either electronically (via a banking institution) or manually via a cheque-run (step B3.8). The invoice to payment process asserts a tight audit trail, which enables AP to know everything about the invoice and payment, supporting the ability to investigate and respond to supplier payment issues (process B3.9). As mentioned above with the non-PO purchasing models, process steps B3.10 - B3.12 provide further details around procurement card purchasing verification and payment processing since it evokes a different payment model - to the bank as opposed to the supplier (since payment has already been made). Finally, in steps B3.13 - B3.15 related vendor data maintenance processes are performed, which play an important role in ensuring correct and proper payment to suppliers, as well as avoiding late charges resulting from incorrect payments (locations, addresses, bank acct #s, etc).

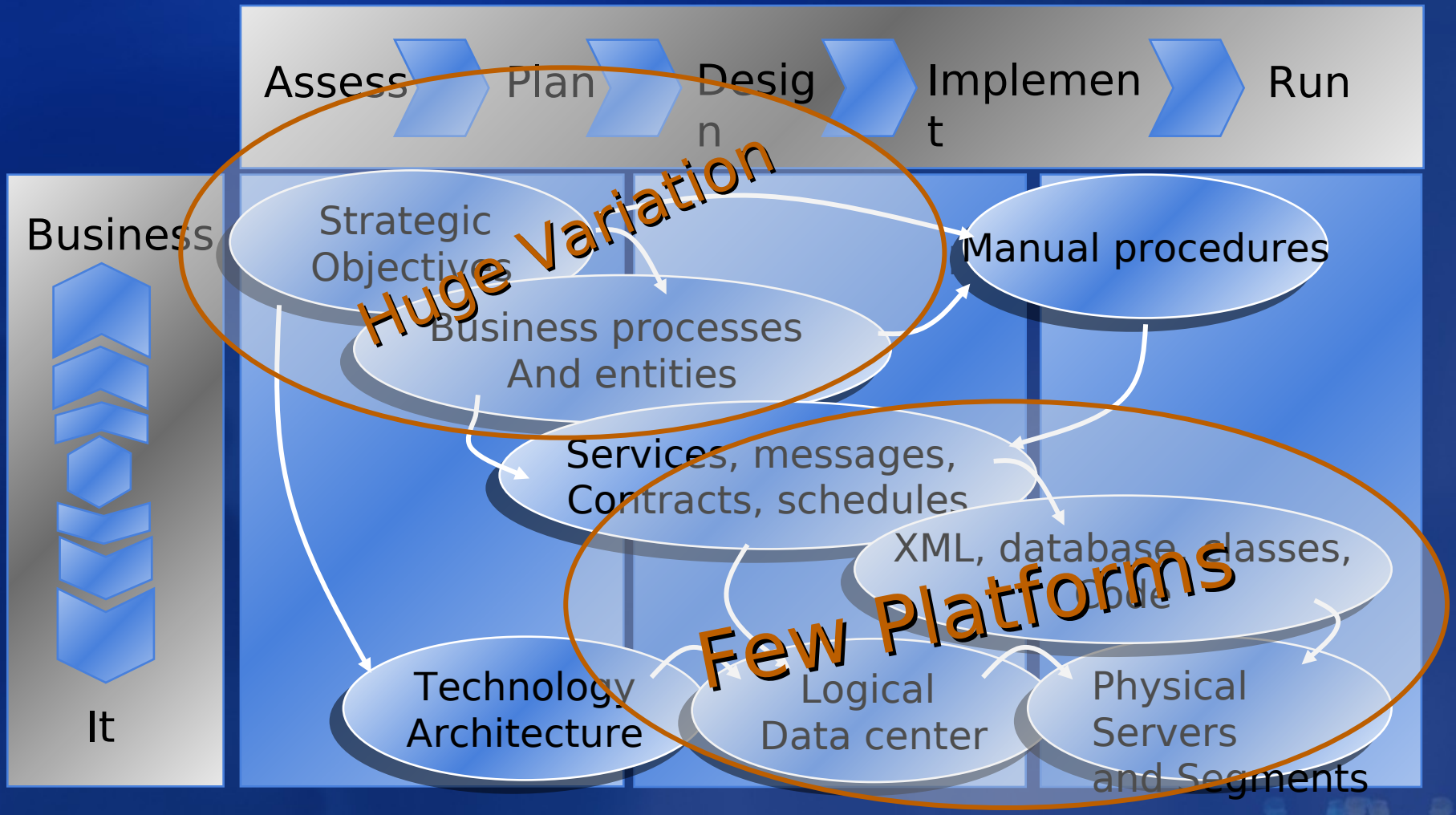
How Do We Describe Business Processes?

- Business Process DSL
  - Activities
  - Business event
  - Dependencies
  - Business entities
  - Documents
  - etc.

Patterns and Transforms

- Process Patterns
  - User process
  - Scheduled process
  - Case process
  - Rules process
- Transformations
  - Transform into Web service collaborations

Build One Business Process DSL?



Building Applications with DSLs

- Choose a specific aspect of overall development
- Define a metamodel for the abstractions
  - OR extend an existing one
- Define a graphical notation
- Define a synchronization engine
- Define Patterns
- Define Aspects

Agenda

- Complexity in Connected Systems
- Using Abstraction to Reduce Complexity
- Domain-Specific Modeling Languages and Patterns
- Scenario 1: Web Services and Design for Operations
- Scenario 2: Implementing Business Processes
- Microsoft Strategy and Tools for Visual Studio “Whidbey”

Making Use of Metadata

- Metadata Collected in DSLs Can Be Made Available Across the Life Cycle
  - Source control systems
  - Debuggers
  - Testing tools
  - Compilers and language editors

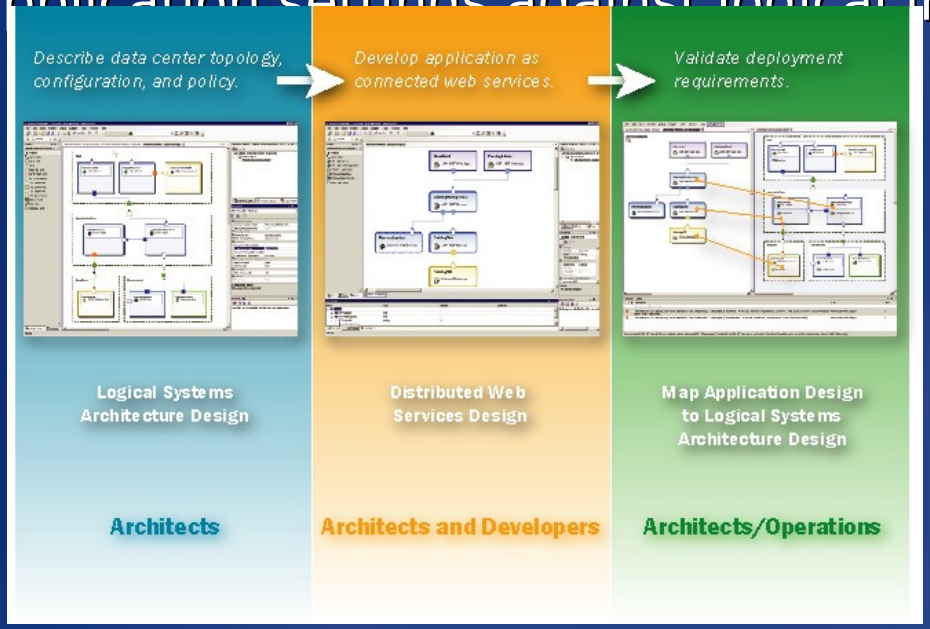
Enterprise Design Tools

***Goal**: Simplify Design and Deployment of Service-Oriented Architectures*

- Build an initial set of model-driven development tools supporting DSLs
  - Focused on specific development tasks
  - With hi-fidelity, continuously synchronized mappings to code or other models
- Expand and integrate model data throughout life cycle tools
- Productize and promote underlying tool-building frameworks and tools

DSLs Supported in “Whidbey”

- Rich support for designing and constructing connected applications from Web services
- Designing Web services for operational deployment
  - Logical infrastructure requirements considered early—inform and constrain developers
  - Verify application settings against logical infrastructure



Review

- Modeling languages can help reduce complexity
  - Domain-specific abstractions
  - Holistic view of scattered concepts
- Microsoft is working to improve Web service design and deployment scenarios
- Implementing DSLs can be made effective with the right underlying engine
- Tool extensibility is very important

Further Information

- White Papers on MSDN®, Visual Studio Enterprise
  - http://msdn.microsoft.com/vstudio/enterprise
- Patterns & Practices on MSDN
  - http://msdn.microsoft.com/practices/type/patterns/

Questions?

Microsoft®